ASTANA IT UNIVERSITY

DIPLOMA PROJECT

DEVELOPMENT AND IMPLEMENTATION OF A BROWSER EXTENSION FOR PHISHING ATTACK PREVENTION

Astana IT University - Cybersecurity

ASTANA IT UNIVERSITY

TEAM MEMBERS

ZHULDYZ MYRZAGALIEVA ASKARKYZY

YERLAN MOLDIR

ARUZHAN RAKHYMZHANOVA ERZHANKYZY

About us

CONTENTS

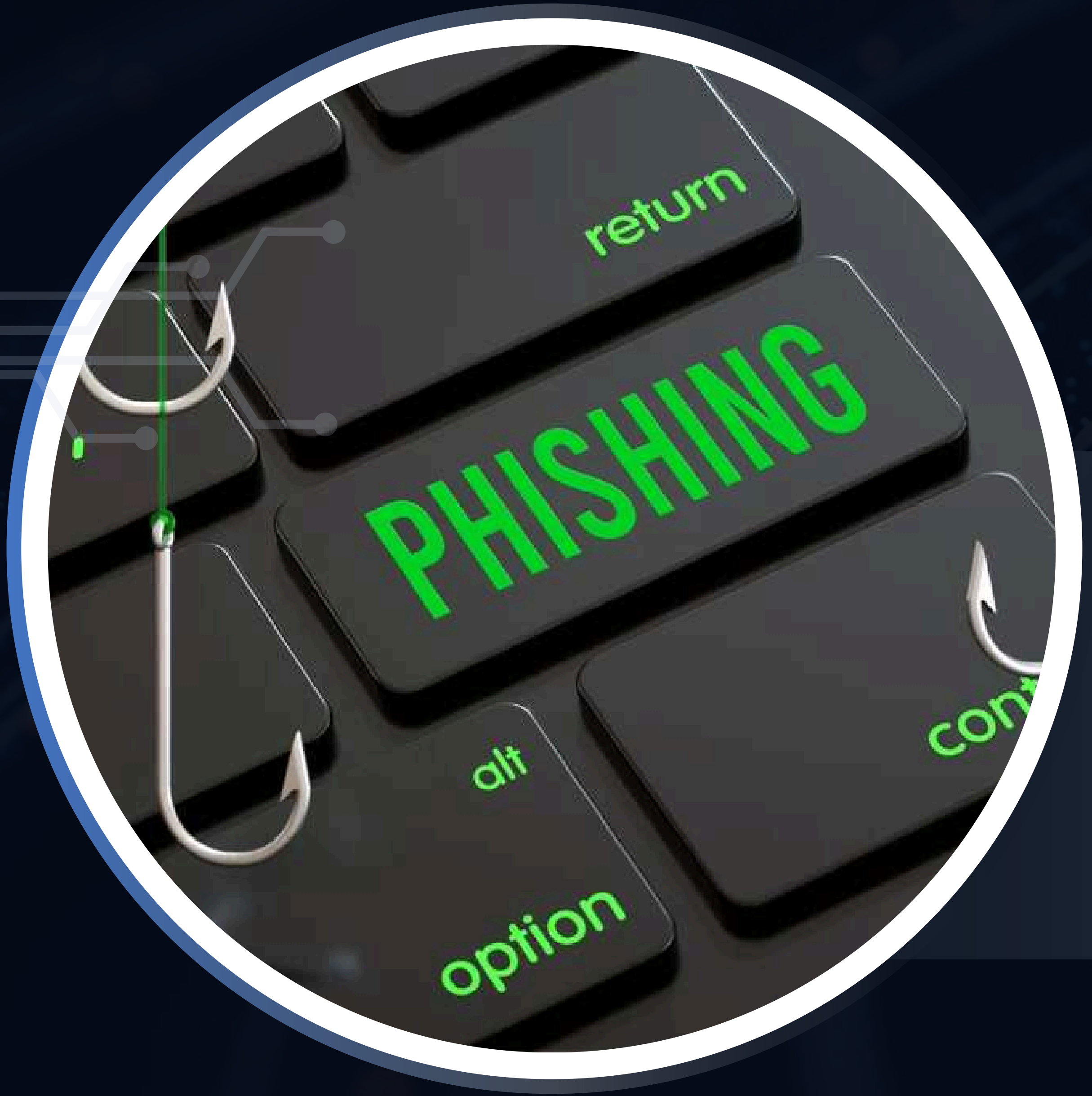
ASTANA IT UNIVERSITY

RELEVANCE OF THE WORK

Phishing attacks are becoming more common and clever, making them a big threat to people's security online. This work is important because it helps create tools to fight against these threats. By making a browser extension that uses comprehensive databases to find phishing websites, this project helps protect user information and improve online security.

ASTANA IT UNIVERSITY

AIM OF THE WORK



The aim of this work is to develop and implement a browser extension for Google Chrome to prevent phishing attacks. This extension will analyze web page content and alert users of potential phishing attempts, enhancing their online security and protecting sensitive information.

OBJECTIVES

Integration with Reliable Databases:

To connect the browser extension with reliable databases to utilize their comprehensive phishing data.

Detecting Phishing in Real-Time:

To set up the extension so it can find phishing websites as soon as users visit them, stopping them from becoming victims.

Easy-to-Use Interface:

To design a simple and easy-to-understand interface so users can easily use the extension.

ASTANA IT UNIVERSITY

LITERATURE REVIEW

Category	Details	Source(s)
Phishing Evolution	From 1990s AOL scams to sophisticated multi-vector phishing campaigns	Gupta et al., 2021
Email Dominance	Over 50% of incidents stem from brand impersonation and BEC (Business Email Compromise) attacks	Verizon, 2023
Mobile Expansion	Threats via SMS, QR codes, and malicious apps targeting users' short attention spans	Chiew et al., 2021
Emerging Vectors	40% rise in social media attacks; growing misuse in AR/VR environments	Kaspersky, 2023; Brown et al., 2023
Blacklist Systems	Fast detection of known threats but vulnerable to zero-day exploits	Sharma et al., 2020
Heuristic Approaches	Detect novel threats with 15–20% false positive rate	Ubing et al., 2021
ML-Based Detection	>95% accuracy in lab tests; limited real-time viability and adversarial resilience	Kumar et al., 2022; Alsajri & AlKahtani, 2024
Browser Extensions	Client-side deployment with rule-based limitations and cross-platform compatibility issues	Varshney et al., 2022

Netcraft Anti-Phishing

- Centralized blocklist, domain age & reputation scoring
- Fast URL checks (<100ms), low false positives (<0.1%)
- ⚠️ Weak on zero-day threats (35–40% detection)

Bitdefender TrafficLight

- Cloud-based URL scan + heuristic content analysis
- High detection of new threats (~89%)
- ⚠️ 15–20% slower load times, ~2.4% false positives

PhishDetect

- Local DOM analysis, no external data sharing
- Strong on template-based phishing
- ⚠️ Limited against evasive, sophisticated attacks

ANALYSIS OF EXISTING SYSTEMS

ASTANA IT UNIVERSITY

PROBLEM STATEMENT

Slow reaction to new phishing tactics:

Many extensions rely on blocklists, making them slow to catch zero-day phishing sites that haven't been reported yet.

Static rule reliance, limited learning ability:

Using fixed detection rules limits the system’s ability to adapt to new and more sophisticated phishing techniques.

Performance issues: slower browsing, high resource use:

Heuristic scanning and content analysis often lead to slower page loads and higher CPU or memory usage during browsing.

ASTANA IT UNIVERSITY

Model Selection and Training

Dataset Analysis and Preparation

Integration with External APIs

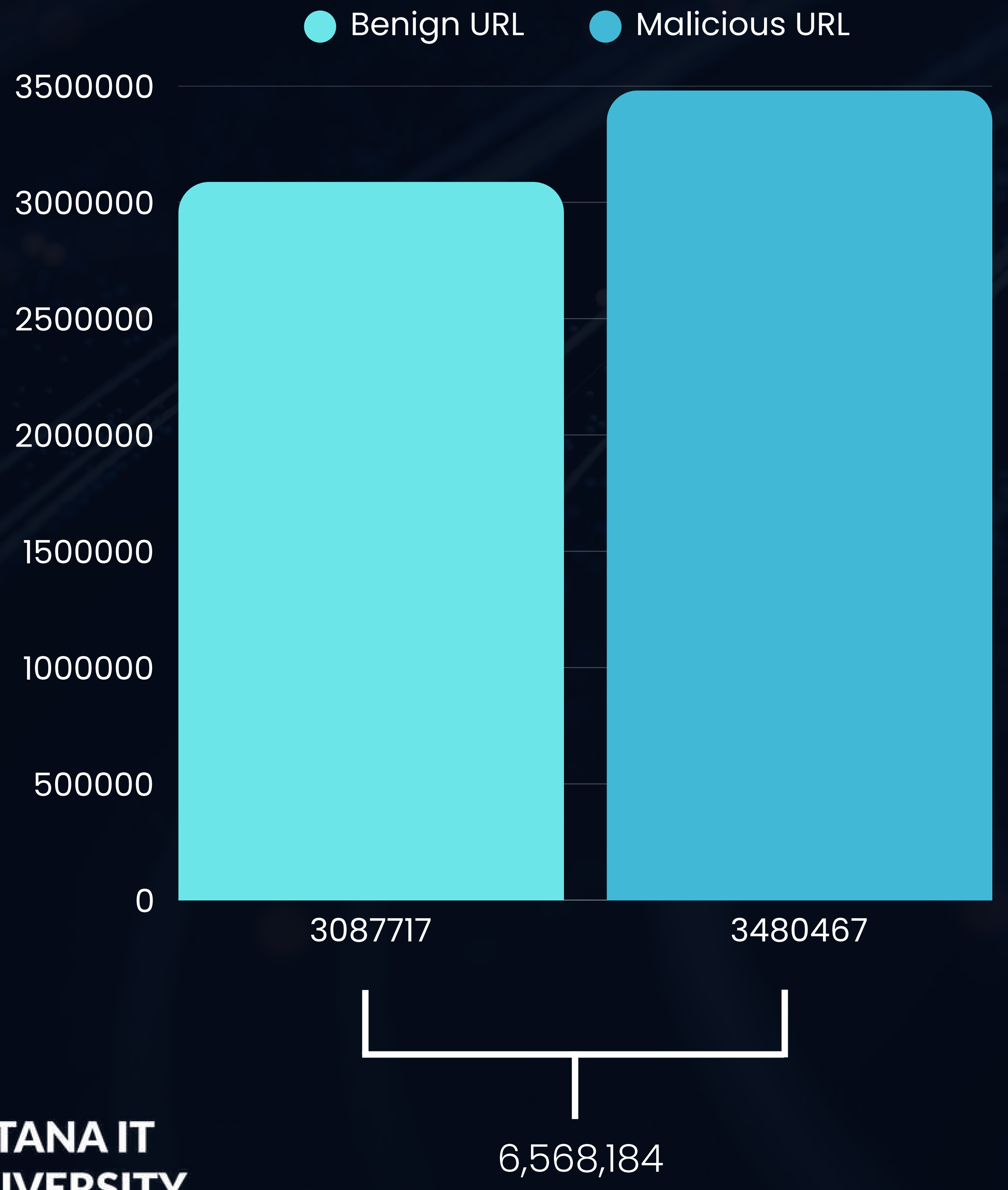
METHODOLOGY

User Feedback Assessment

DOM Analysis of Web Pages

System Architecture

DATASET ANALYSIS AND PREPARATION



domain_entropy, domain_length, has_anchor, has_digits_in_domain, has_https, has_ip, has_port, has_query, is_shortening_service, num_and, num_asterisks, num_at, num_commas, num_dollars, num_dots, num_equals, num_exclamation, num_hashes, num_hyphens, num_percent, num_plus, num_question_marks, num_slashes, num_spaces, num_special_chars, num_subdomains, num_tildes, num_underscores, path_entropy, path_length, query_entropy, query_length, suspicious_tld, suspicious_words_count, url_length

34 URL Features extracted by

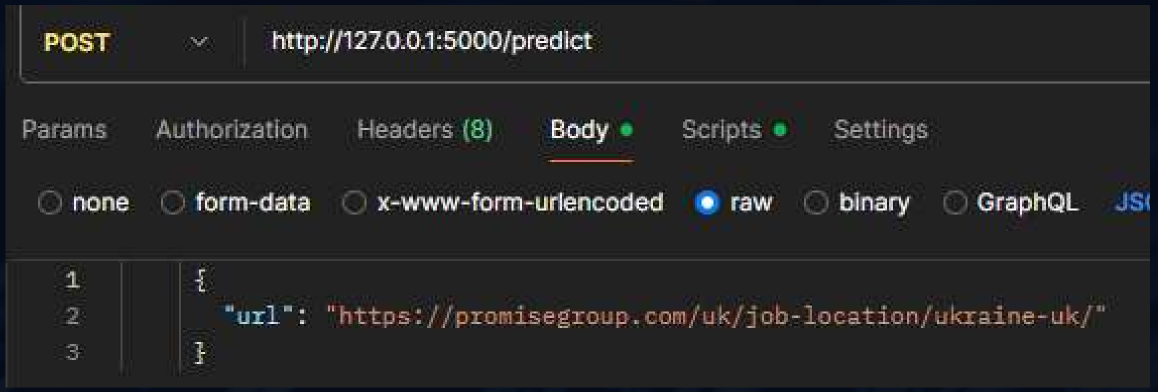
`extract_features.py`

ASTANA IT UNIVERSITY

FEATURE EXTRACTION FUNCTION (PYTHON)

```python
features = {
    'url_length': len(url_without_protocol),
    'domain_length': len(domain),
    'num_dots': url_without_protocol.count('.'),
    'num_slashes': url_without_protocol.count('/'),
    'num_hyphens': url_without_protocol.count('-'),
    'num_underscores': url_without_protocol.count('_'),
    'num_question_marks': url_without_protocol.count('?'),
    'num_equals': url_without_protocol.count('='),
    'num_at': url_without_protocol.count('@'),
    'num_and': url_without_protocol.count('&'),
    'num_exclamation': url_without_protocol.count('!'),
    'num_spaces': url_without_protocol.count(' '),
    'num_tildes': url_without_protocol.count('~'),
    'num_commas': url_without_protocol.count(','),
    'num_plus': url_without_protocol.count('+'),
    'num_asterisks': url_without_protocol.count('*'),
    'num_hashes': url_without_protocol.count('#'),
    'num_dollars': url_without_protocol.count('$'),
    'num_percent': url_without_protocol.count('%'),
    'num_special_chars': sum(url_without_protocol.count(c) for c in '!@#$%^&*()_+-=[]{}|;:,.<>?/~'),
    'has_https': int(parsed.scheme == 'https'),
    'has_ip': int(bool(re.search(r'(\d{1,3}\.){3}\d{1,3}', url_without_protocol))),
    'has_port': int(':' in domain),
    'has_query': int(bool(query)),
    'has_anchor': int('#' in url_without_protocol),
    'has_digits_in_domain': int(bool(re.search(r'\d', domain_name))),
    'suspicious_tld': int(tld in suspicious_tlds),
    'num_subdomains': len(ext.subdomain.split('.')) if ext.subdomain else 0,
    'is_shortening_service': int(any(service in domain for service in shortening_services)),
    'suspicious_words_count': sum(1 for word in suspicious_words if word in url_without_protocol.lower()),
    'path_length': len(path),
    'query_length': len(query),
    'domain_entropy': -sum((count/len(domain)) * np.log2(count/len(domain))
                          for count in Counter(domain).values()) if domain else 0,
    'path_entropy': -sum((count/len(path)) * np.log2(count/len(path))
                        for count in Counter(path).values()) if path else 0,
    'query_entropy': -sum((count/len(query)) * np.log2(count/len(query))
                         for count in Counter(query).values()) if query else 0
}

return features
```

API REQUEST EXAMPLE

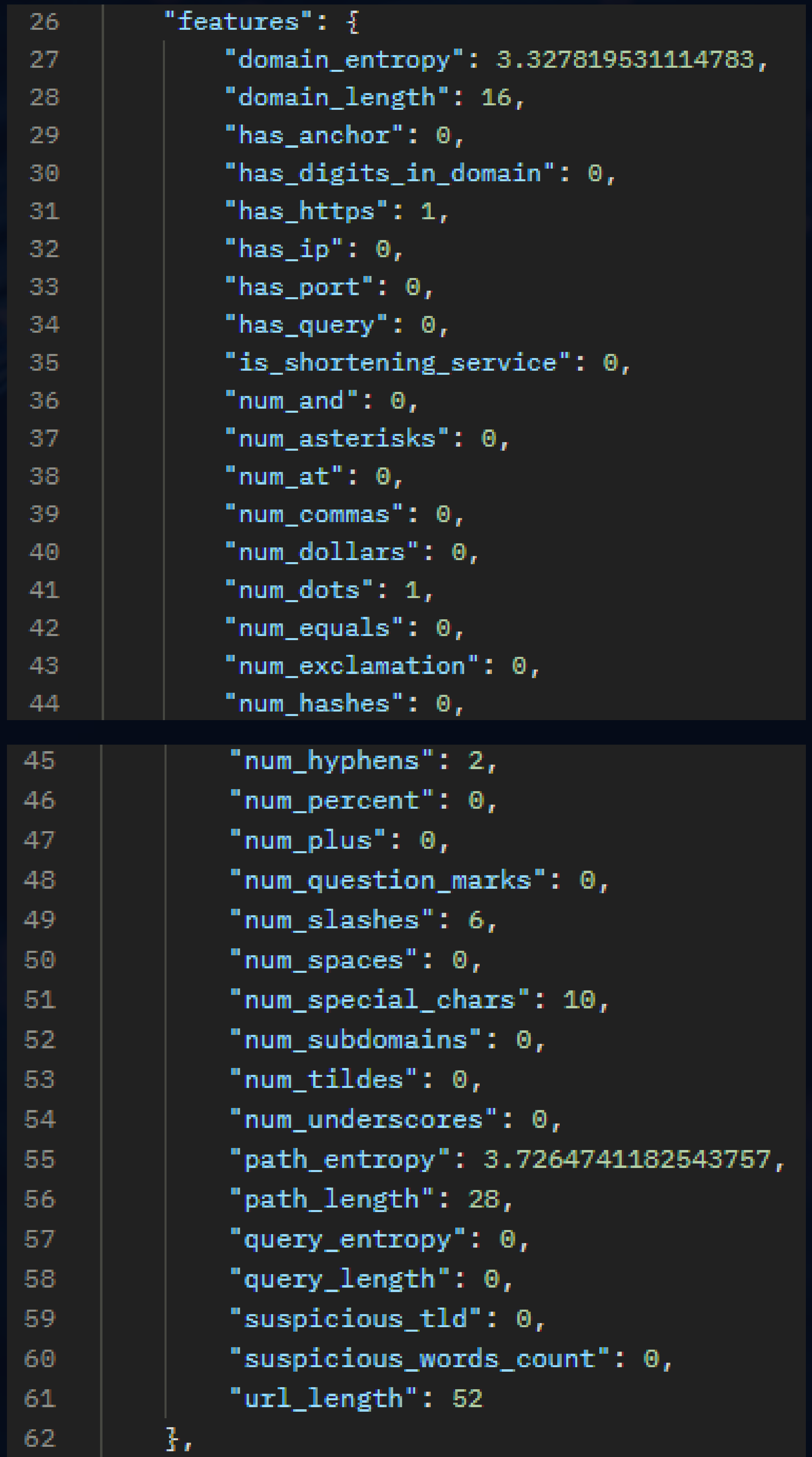


POST http://127.0.0.1:5000/predict

Body: [✓ raw]

```json
{
  "url": "https://promisegroup.com/uk/job-location/ukraine-uk/"
}
```

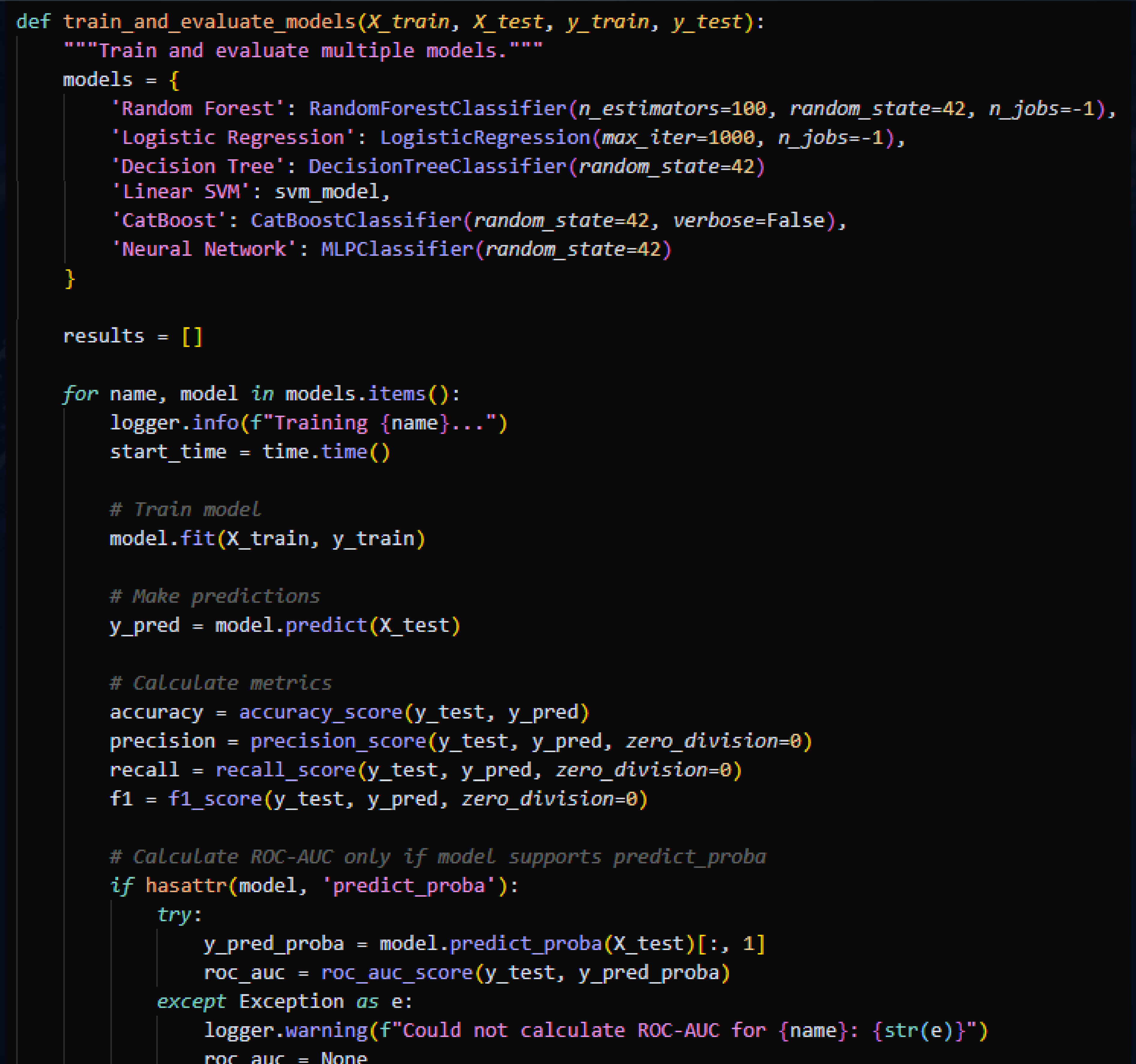
EXTRACTED FEATURES (JSON RESPONSE)



```json
    "features": {
        "domain_entropy": 3.327819531114783,
        "domain_length": 16,
        "has_anchor": 0,
        "has_digits_in_domain": 0,
        "has_https": 1,
        "has_ip": 0,
        "has_port": 0,
        "has_query": 0,
        "is_shortening_service": 0,
        "num_and": 0,
        "num_asterisks": 0,
        "num_at": 0,
        "num_commas": 0,
        "num_dollars": 0,
        "num_dots": 1,
        "num_equals": 0,
        "num_exclamation": 0,
        "num_hashes": 0,
        "num_hyphens": 2,
        "num_percent": 0,
        "num_plus": 0,
        "num_question_marks": 0,
        "num_slashes": 6,
        "num_spaces": 0,
        "num_special_chars": 10,
        "num_subdomains": 0,
        "num_tildes": 0,
        "num_underscores": 0,
        "path_entropy": 3.7264741182543757,
        "path_length": 28,
        "query_entropy": 0,
        "query_length": 0,
        "suspicious_tld": 0,
        "suspicious_words_count": 0,
        "url_length": 52
    },
```

ASTANA IT UNIVERSITY

MODELS TRAINING

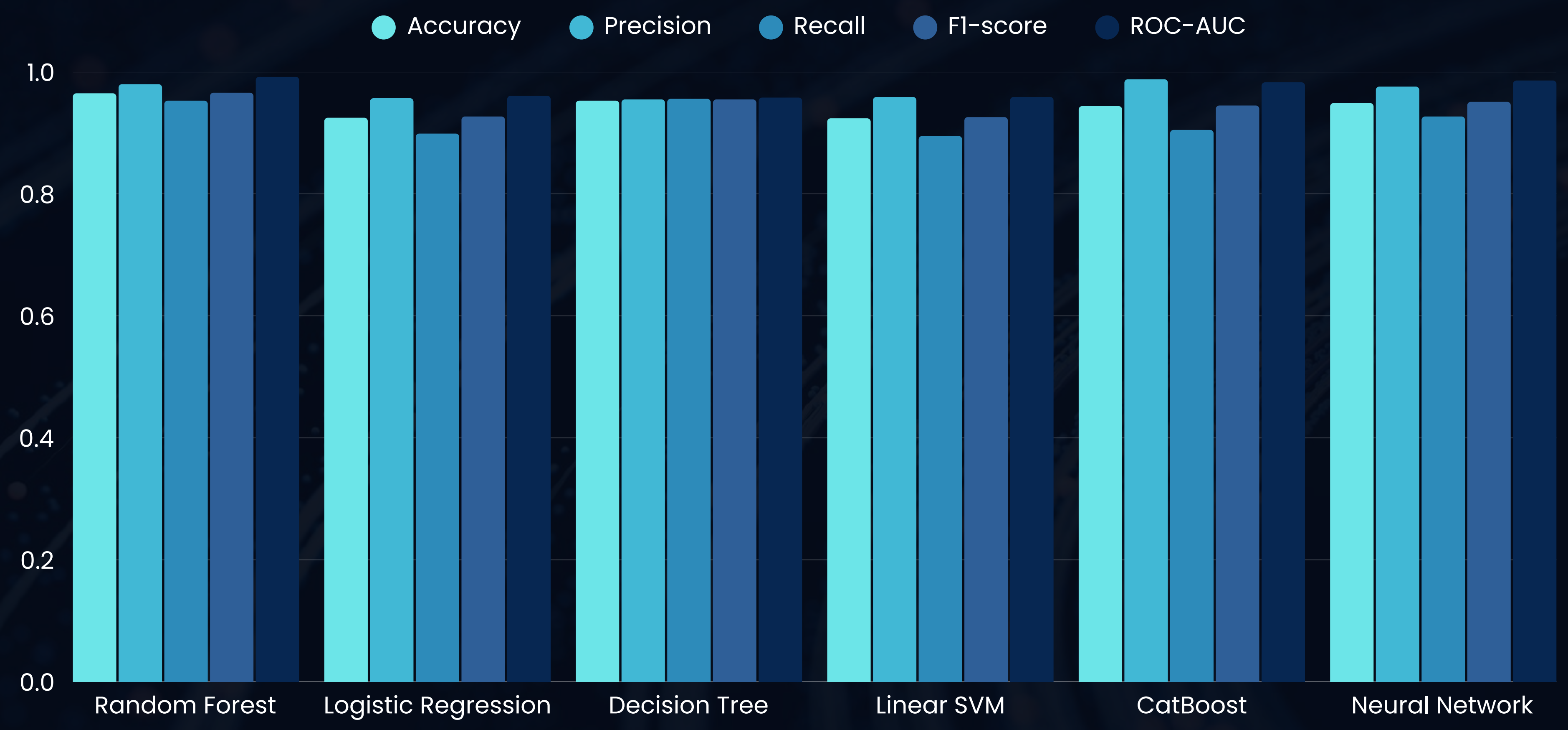


```
def train_and_evaluate_models(X_train, X_test, y_train, y_test):
    """Train and evaluate multiple models."""
    models = {
        'Random Forest': RandomForestClassifier(n_estimators=100, random_state=42, n_jobs=-1),
        'Logistic Regression': LogisticRegression(max_iter=1000, n_jobs=-1),
        'Decision Tree': DecisionTreeClassifier(random_state=42)
        'Linear SVM': svm_model,
        'CatBoost': CatBoostClassifier(random_state=42, verbose=False),
        'Neural Network': MLPClassifier(random_state=42)
    }

    results = []

    for name, model in models.items():
        logger.info(f"Training {name}...")
        start_time = time.time()

        # Train model
        model.fit(X_train, y_train)

        # Make predictions
        y_pred = model.predict(X_test)

        # Calculate metrics
        accuracy = accuracy_score(y_test, y_pred)
        precision = precision_score(y_test, y_pred, zero_division=0)
        recall = recall_score(y_test, y_pred, zero_division=0)
        f1 = f1_score(y_test, y_pred, zero_division=0)

        # Calculate ROC-AUC only if model supports predict_proba
        if hasattr(model, 'predict_proba'):
            try:
                y_pred_proba = model.predict_proba(X_test)[:, 1]
                roc_auc = roc_auc_score(y_test, y_pred_proba)
            except Exception as e:
                logger.warning(f"Could not calculate ROC-AUC for {name}: {str(e)}")
                roc_auc = None
```

ASTANA IT UNIVERSITY

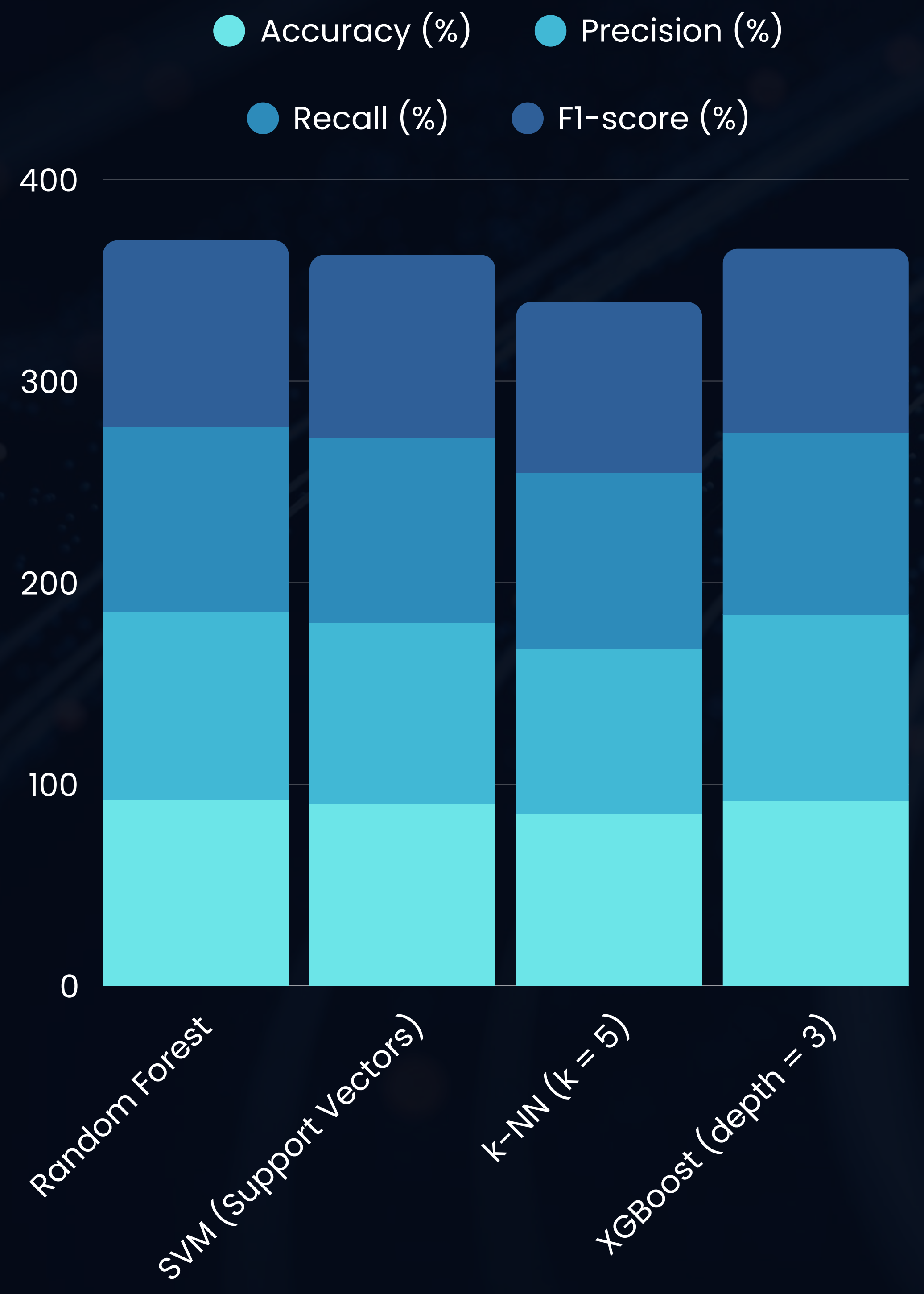
MODEL SELECTION

ASTANA IT UNIVERSITY



Model	Accuracy	Precision	Recall	F1-score	ROC-AUC	Train Time (s)
Random Forest	**0.965**	**0.98**	**0.953**	**0.966**	**0.992**	142.2
Logistic Regression	0.925	0.957	0.899	0.927	0.961	44.89
Decision Tree	0.953	0.955	0.956	0.955	0.958	50.07
Linear SVM	0.924	0.959	0.895	0.926	0.959	804.23
CatBoost	0.944	0.988	0.905	0.945	0.983	90
Neural Network	0.949	0.976	0.927	0.951	**0.986**	**1243.74**

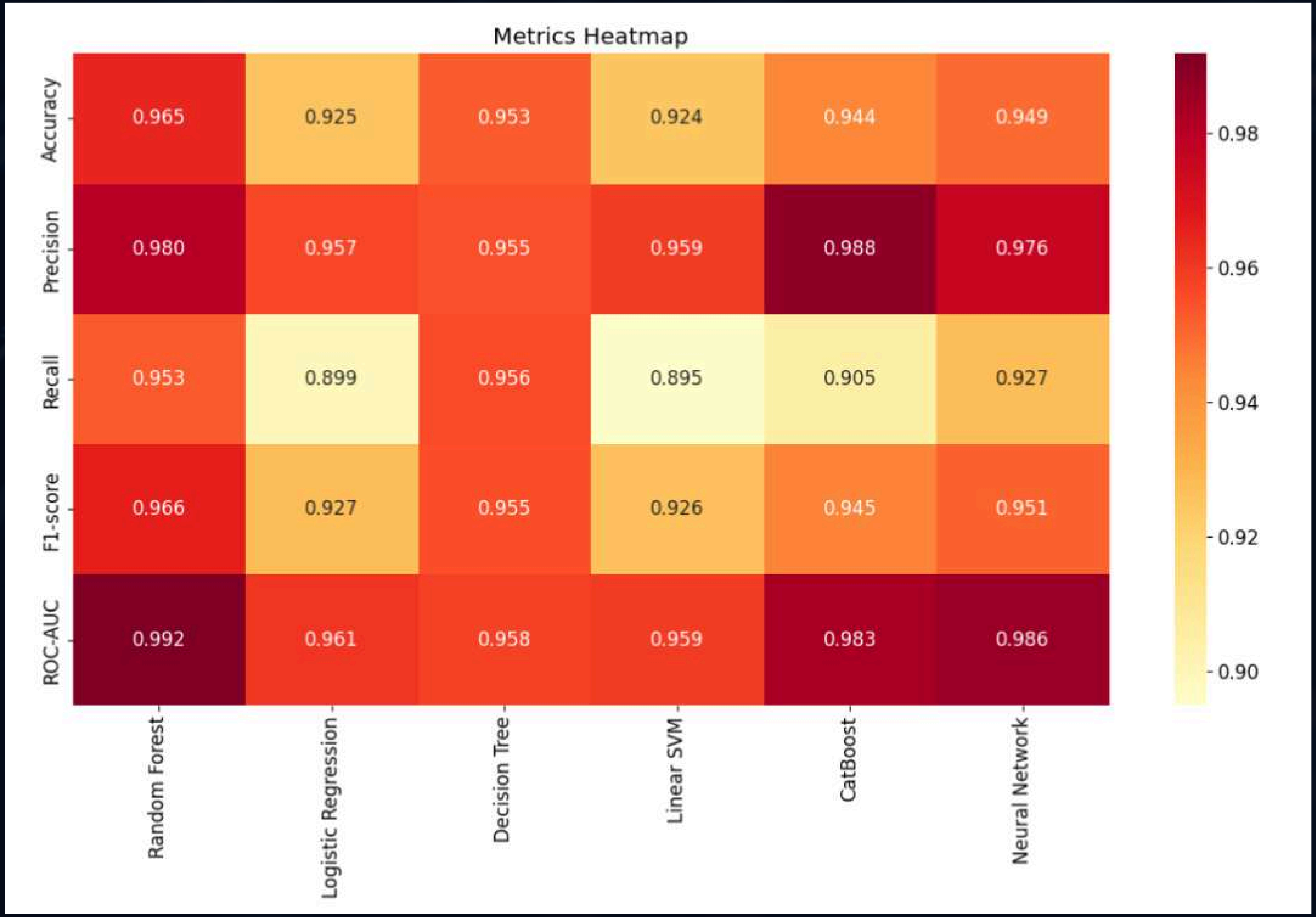
ADDITIONAL MODEL TRAINING AND RESULTS



Model	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
Random Forest	92.4	93	92	92.5
SVM (Support Vectc	90.5	89.7	91.8	90.7
k-NN (k = 5)	85	82.1	87.5	84.7
XGBoost (depth = 3)	91.7	92.5	90.2	91.3

ASTANA IT UNIVERSITY

MODEL SELECTION AND RESULTS



	Random Forest	Logistic Regression	Decision Tree	Linear SVM	CatBoost	Neural Network
Accuracy	0.965	0.925	0.953	0.924	0.944	0.949
Precision	0.980	0.957	0.955	0.959	0.988	0.976
Recall	0.953	0.899	0.956	0.895	0.905	0.927
F1-score	0.966	0.927	0.955	0.926	0.945	0.951
ROC-AUC	0.992	0.961	0.958	0.959	0.983	0.986

ASTANA IT UNIVERSITY

EXTERNAL API AND DOM ANALYSIS INTEGRATION

ML Model + VirusTotal Google Safe Browsing + DOM analysis

ASTANA IT UNIVERSITY

EXTERNAL API AND DOM ANALYSIS INTEGRATION

VirusTotal

```python
import requests

def check_url_virustotal(url, api_key):
    vt_url = "https://www.virustotal.com/api/v3/urls"
    headers = {"x-apikey": api_key}
    response = requests.post(vt_url, data={"url": url}, headers=headers)
    if response.status_code == 200:
        analysis_id = response.json()["data"]["id"]
        # Получить результат анализа
        result = requests.get(f"{vt_url}/{analysis_id}", headers=headers)
        return result.json()
    return None
```

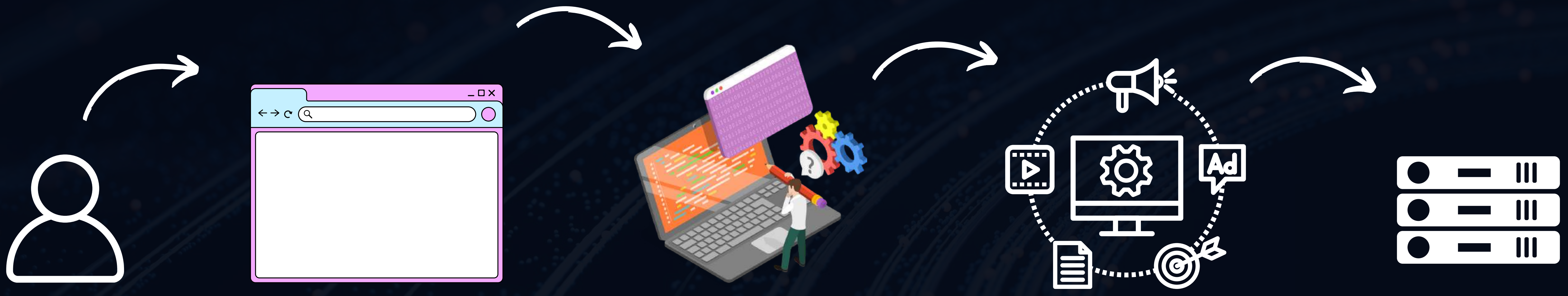
Google Safe Browsing

```python
import requests

def check_url_gsb(url, api_key):
    gsb_url = "https://safebrowsing.googleapis.com/v4/threatMatches:find"
    payload = {
        "client": {"clientId": "your-app", "clientVersion": "1.0"},
        "threatInfo": {
            "threatTypes": ["MALWARE", "SOCIAL_ENGINEERING"],
            "platformTypes": ["ANY_PLATFORM"],
            "threatEntryTypes": ["URL"],
            "threatEntries": [{"url": url}]
        }
    }
    params = {"key": api_key}
    response = requests.post(gsb_url, params=params, json=payload)
    return response.json()
```

DOM analysis

```javascript
// Check for sensitive keywords that might indicate phishing
function checkForSensitiveKeywords(content) {
  return CONFIG.SENSITIVE_KEYWORDS.some(keyword =>
    content.includes(keyword.toLowerCase())
  );
}

// Check for suspicious domain patterns
function checkSuspiciousDomain(domain) {
  return CONFIG.SUSPICIOUS_DOMAINS.some(pattern =>
    domain.includes(pattern)
  );
}

// Check for suspicious form patterns
function checkSuspiciousForms() {
  const forms = document.querySelectorAll('form');
  const suspiciousPatterns = [
    'action="http://"',
    'action="https://"',
    'method="get"',
    'autocomplete="off"'
  ];

  for (const form of forms) {
    const formHtml = form.outerHTML.toLowerCase();
    if (suspiciousPatterns.some(pattern => formHtml.includes(pattern))) {
      return true;
    }
  }
  return false;
}

// Check for external links in forms
function checkExternalLinks() {
  const forms = document.querySelectorAll('form');
  const currentDomain = window.location.hostname;

  for (const form of forms) {
    if (form.action && !form.action.includes(currentDomain)) {
      return true;
    }
  }
  return false;
}
```

SYSTEM ARCHITECTURE



User

Popup.js

handleUserInput()
chrome.runtime.sendMessage(...)

Background.js

fetch(...),
chrome.runtime.onMessage.addListener(...)

Content.js

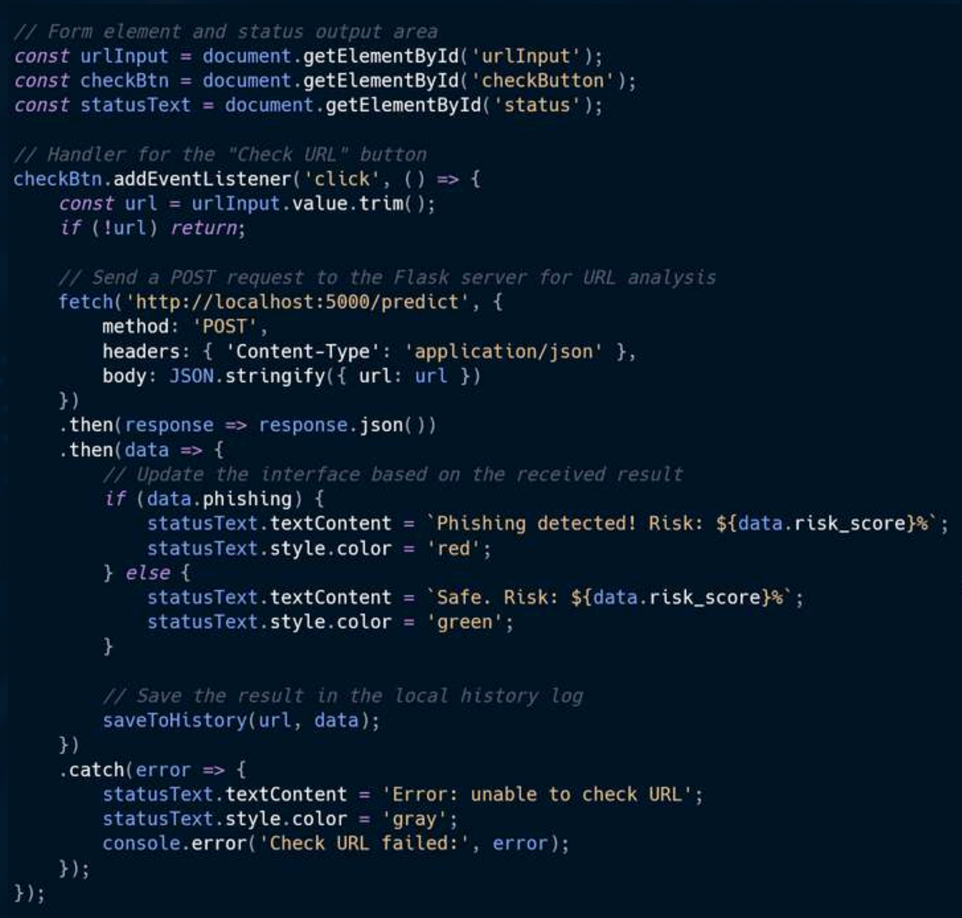
analyzePageContent()

Server: app.py

@app.route('/predict', methods=['POST'])
extract_url_features(url)
model.predict_proba()
check_url_virustotal(url),
check_url_gsb(url)

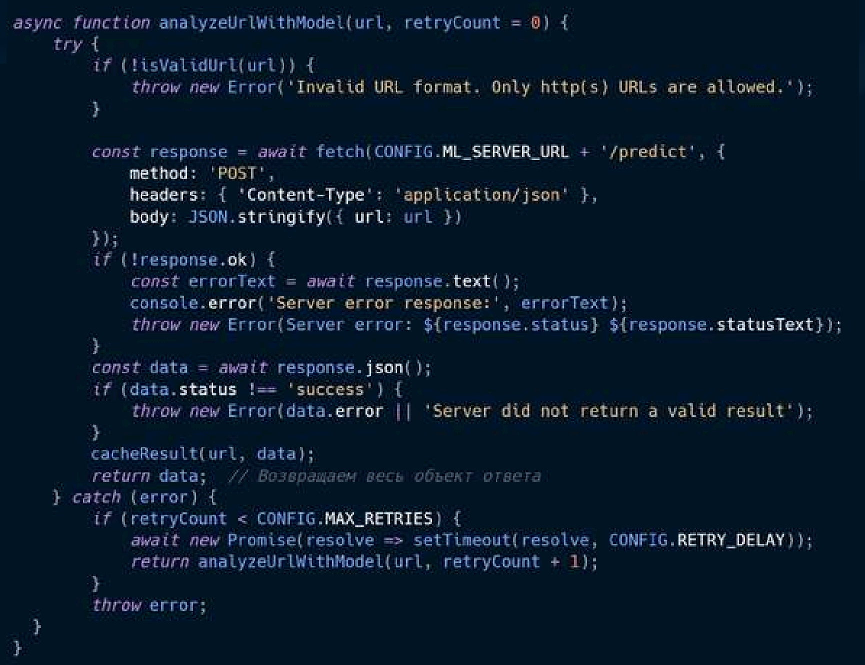
ASTANA IT
UNIVERSITY

SYSTEM ARCHITECTURE: CODE REVIEW

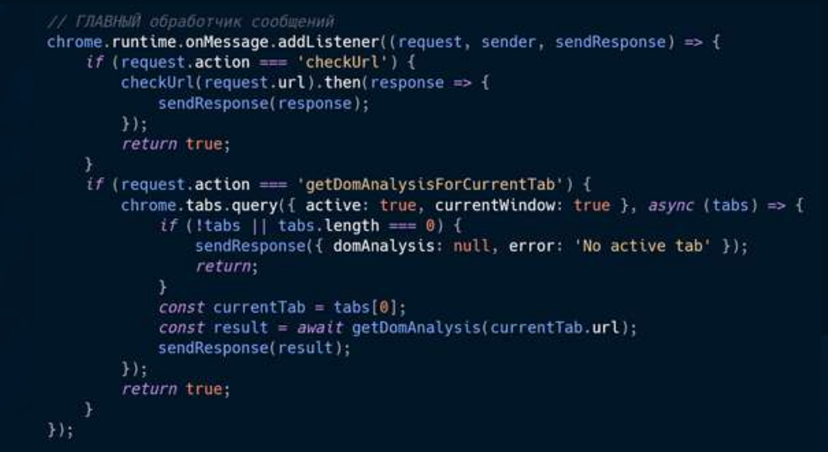


```javascript
// Form element and status output area
const urlInput = document.getElementById('urlInput');
const checkBtn = document.getElementById('checkButton');
const statusText = document.getElementById('status');

// Handler for the "Check URL" button
checkBtn.addEventListener('click', () => {
    const url = urlInput.value.trim();
    if (!url) return;

    // Send a POST request to the Flask server for URL analysis
    fetch('http://localhost:5000/predict', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ url: url })
    })
    .then(response => response.json())
    .then(data => {
        // Update the interface based on the received result
        if (data.phishing) {
            statusText.textContent = `Phishing detected! Risk: ${data.risk_score}%`;
            statusText.style.color = 'red';
        } else {
            statusText.textContent = `Safe. Risk: ${data.risk_score}%`;
            statusText.style.color = 'green';
        }

        // Save the result in the local history log
        saveToHistory(url, data);
    })
    .catch(error => {
        statusText.textContent = 'Error: unable to check URL';
        statusText.style.color = 'gray';
        console.error('Check URL failed:', error);
    });
});
```

Fragment of popup.js: Handling Manual URL Check and Displaying the Result

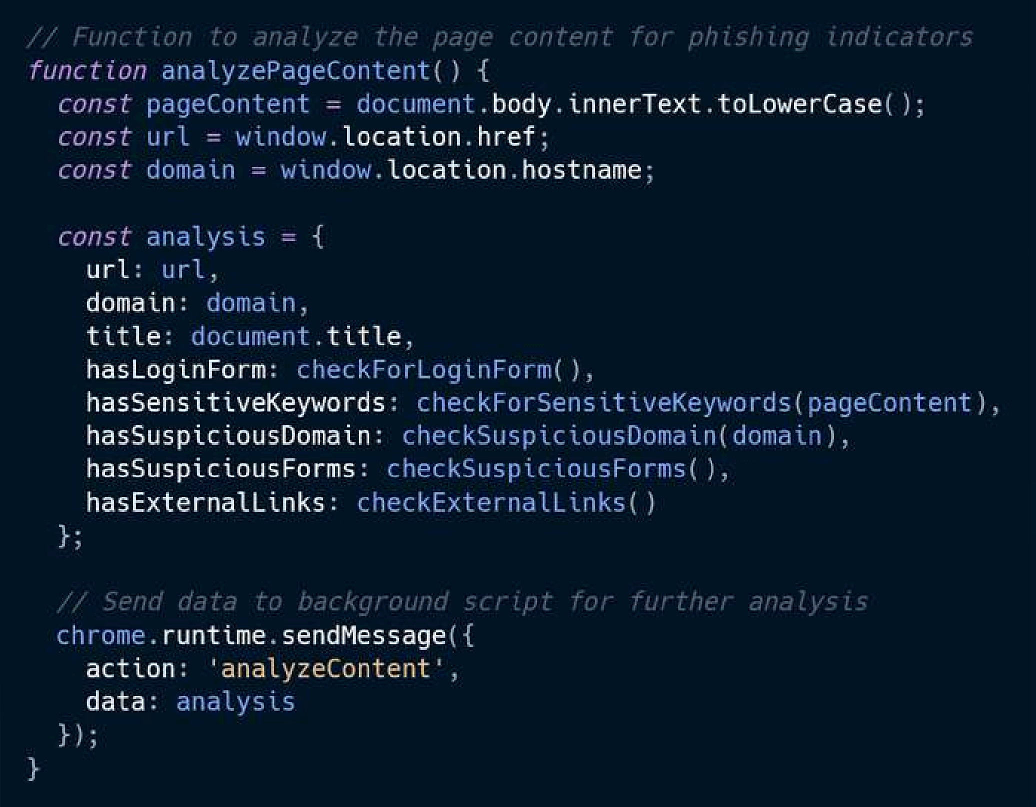


```javascript
async function analyzeUrlWithModel(url, retryCount = 0) {
    try {
        if (!isValidUrl(url)) {
            throw new Error('Invalid URL format. Only http(s) URLs are allowed.');
        }

        const response = await fetch(CONFIG.ML_SERVER_URL + '/predict', {
            method: 'POST',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify({ url: url })
        });
        if (!response.ok) {
            const errorText = await response.text();
            console.error('Server error response:', errorText);
            throw new Error(Server error: ${response.status} ${response.statusText});
        }
        const data = await response.json();
        if (data.status !== 'success') {
            throw new Error(data.error || 'Server did not return a valid result');
        }
        cacheResult(url, data);
        return data;  // Возвращаем весь объект ответа
    } catch (error) {
        if (retryCount < CONFIG.MAX_RETRIES) {
            await new Promise(resolve => setTimeout(resolve, CONFIG.RETRY_DELAY));
            return analyzeUrlWithModel(url, retryCount + 1);
        }
        throw error;
    }
}
```



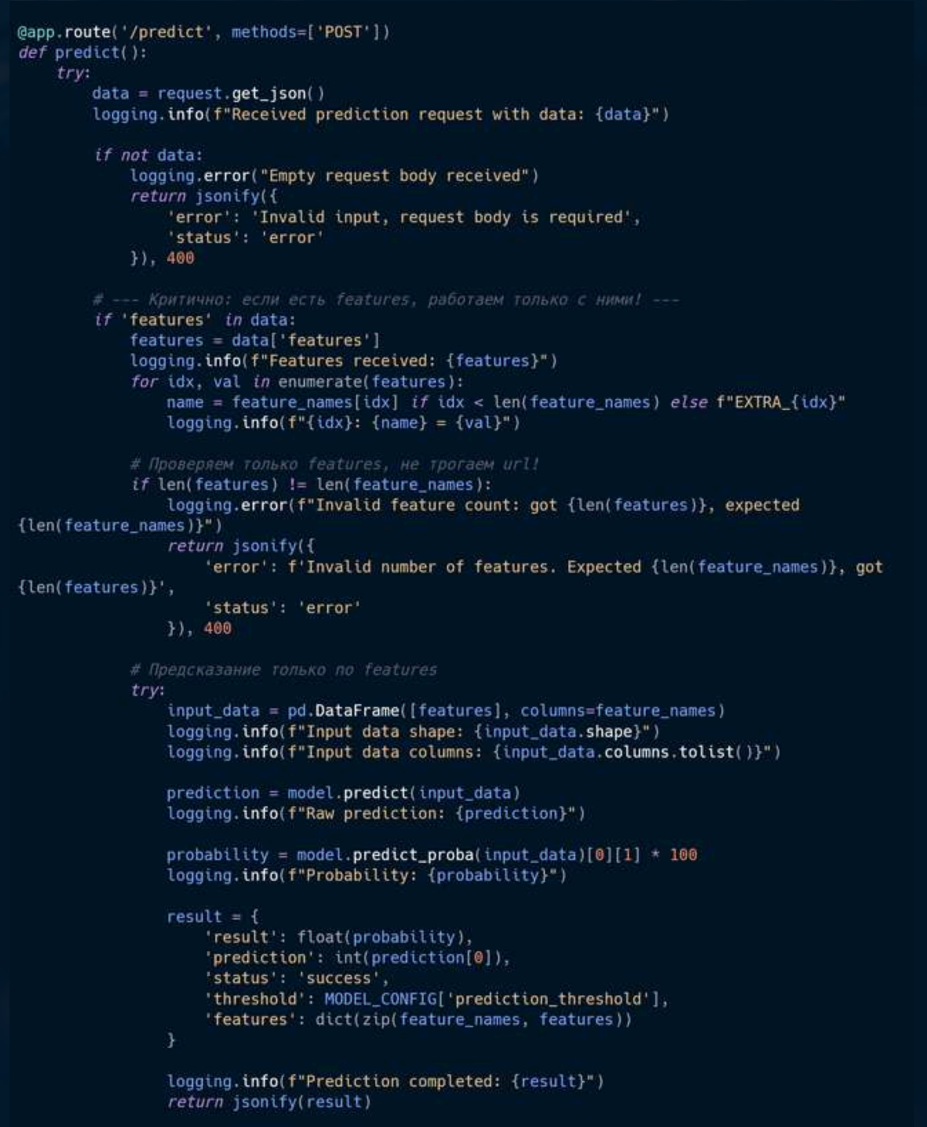
```javascript
// ГЛАВНЫЙ обработчик сообщений
chrome.runtime.onMessage.addListener((request, sender, sendResponse) => {
    if (request.action === 'checkUrl') {
        checkUrl(request.url).then(response => {
            sendResponse(response);
        });
        return true;
    }
    if (request.action === 'getDomAnalysisForCurrentTab') {
        chrome.tabs.query({ active: true, currentWindow: true }, async (tabs) => {
            if (!tabs || tabs.length === 0) {
                sendResponse({ domAnalysis: null, error: 'No active tab' });
                return;
            }
            const currentTab = tabs[0];
            const result = await getDomAnalysis(currentTab.url);
            sendResponse(result);
        });
        return true;
    }
});
```

Fragment of background.js: Handling Messages and Sending URL for Server-Side Analysis



```javascript
// Function to analyze the page content for phishing indicators
function analyzePageContent() {
  const pageContent = document.body.innerText.toLowerCase();
  const url = window.location.href;
  const domain = window.location.hostname;

  const analysis = {
    url: url,
    domain: domain,
    title: document.title,
    hasLoginForm: checkForLoginForm(),
    hasSensitiveKeywords: checkForSensitiveKeywords(pageContent),
    hasSuspiciousDomain: checkSuspiciousDomain(domain),
    hasSuspiciousForms: checkSuspiciousForms(),
    hasExternalLinks: checkExternalLinks()
  };

  // Send data to background script for further analysis
  chrome.runtime.sendMessage({
    action: 'analyzeContent',
    data: analysis
  });
}
```

Fragment of content.js: Sending the Current URL for Analysis and Responding to a Dangerous Site



```python
@app.route('/predict', methods=['POST'])
def predict():
    try:
        data = request.get_json()
        logging.info(f"Received prediction request with data: {data}")

        if not data:
            logging.error("Empty request body received")
            return jsonify({
                'error': 'Invalid input, request body is required',
                'status': 'error'
            }), 400

        # --- Критично: если есть features, работаем только с ними! ---
        if 'features' in data:
            features = data['features']
            logging.info(f"Features received: {features}")
            for idx, val in enumerate(features):
                name = feature_names[idx] if idx < len(feature_names) else f"EXTRA_{idx}"
                logging.info(f"{idx}: {name} = {val}")

            # Проверяем только features, не трогаем url!
            if len(features) != len(feature_names):
                logging.error(f"Invalid feature count: got {len(features)}, expected
{len(feature_names)}")
                return jsonify({
                    'error': f'Invalid number of features. Expected {len(feature_names)}, got
{len(features)}',
                    'status': 'error'
                }), 400

            # Предсказание только по features
            try:
                input_data = pd.DataFrame([features], columns=feature_names)
                logging.info(f"Input data shape: {input_data.shape}")
                logging.info(f"Input data columns: {input_data.columns.tolist()}")

                prediction = model.predict(input_data)
                logging.info(f"Raw prediction: {prediction}")

                probability = model.predict_proba(input_data)[0][1] * 100
                logging.info(f"Probability: {probability}")

                result = {
                    'result': float(probability),
                    'prediction': int(prediction[0]),
                    'status': 'success',
                    'threshold': MODEL_CONFIG['prediction_threshold'],
                    'features': dict(zip(feature_names, features))
                }

                logging.info(f"Prediction completed: {result}")
                return jsonify(result)
```

Fragment from app.py: Initialization of the Flask Application and Loading of the Pre-trained Machine Learning Model

USER FEEDBACK ASSESSMENT

30 answers

12 closed question and
3 open questions

ASTANA IT UNIVERSITY

SUMMARY OF SURVEY RESULTS

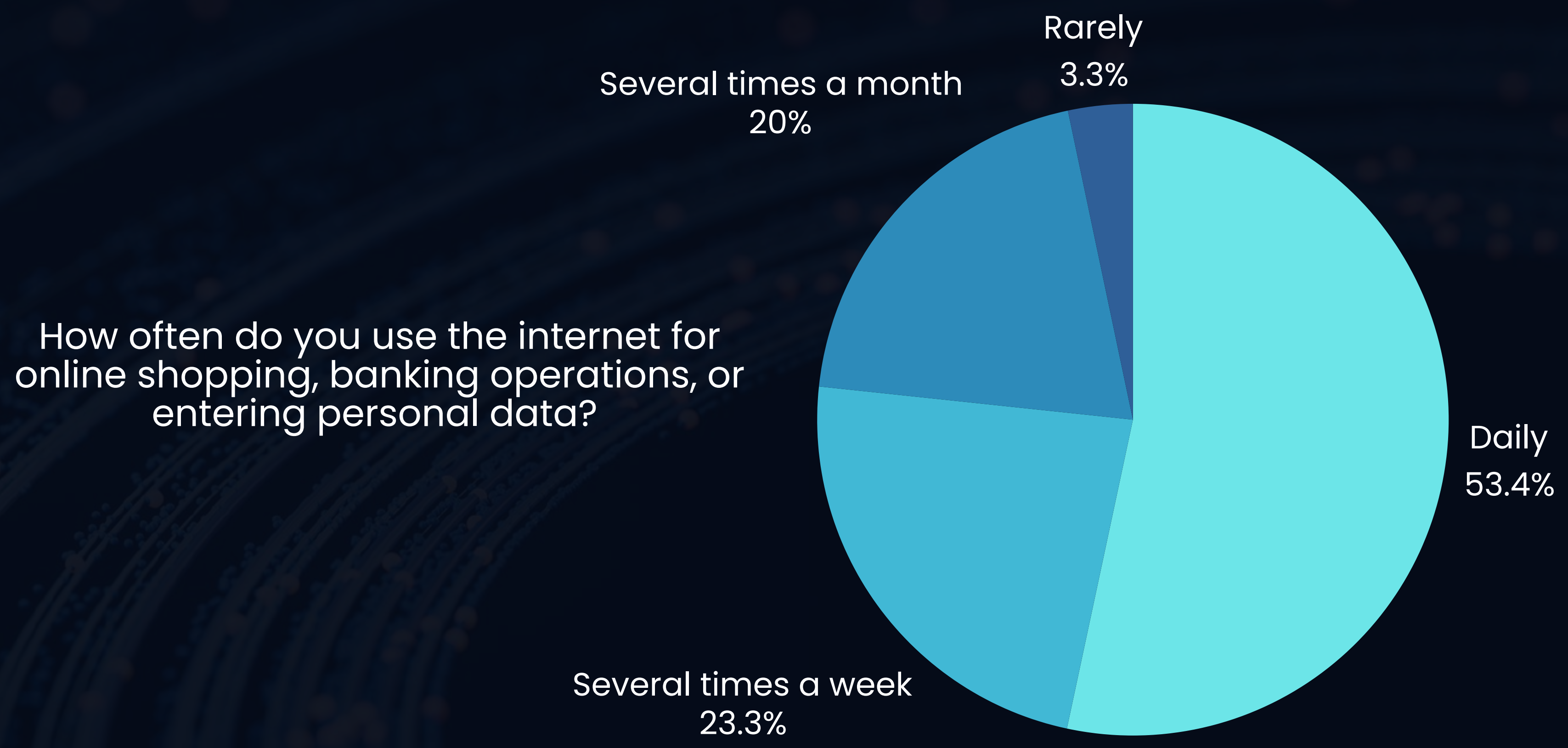
High level of digital activity among the audience

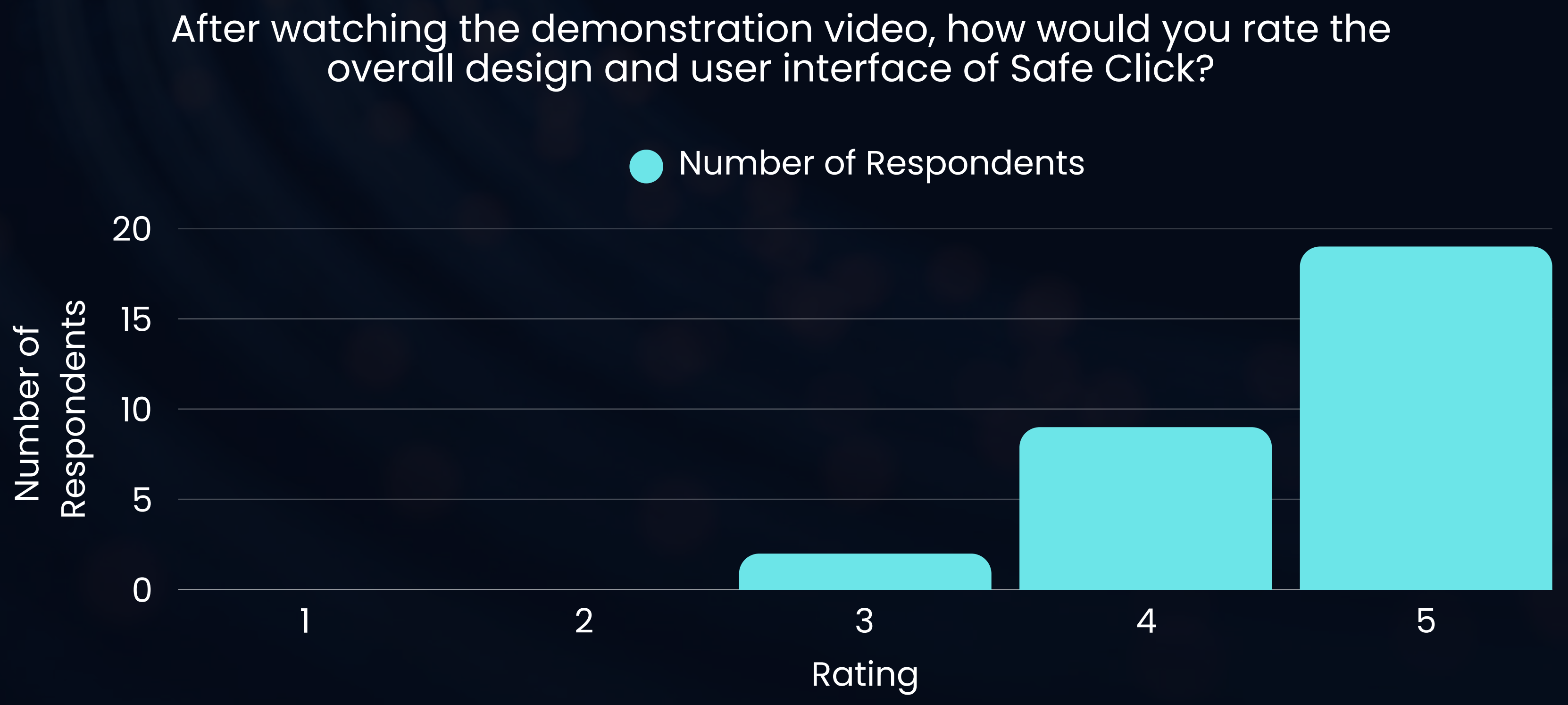
Over 90% of respondents regularly perform actions online that involve phishing risks, such as payments, logins, or sharing sensitive information.

Interface is considered clear and user-friendly

The vast majority rated the interface 4 or 5 out of 5 (93.3%).

The risk score system (numeric value + color indicator) is perceived as intuitive and easy to interpret.





ASTANA IT UNIVERSITY

SUMMARY OF SURVEY RESULTS

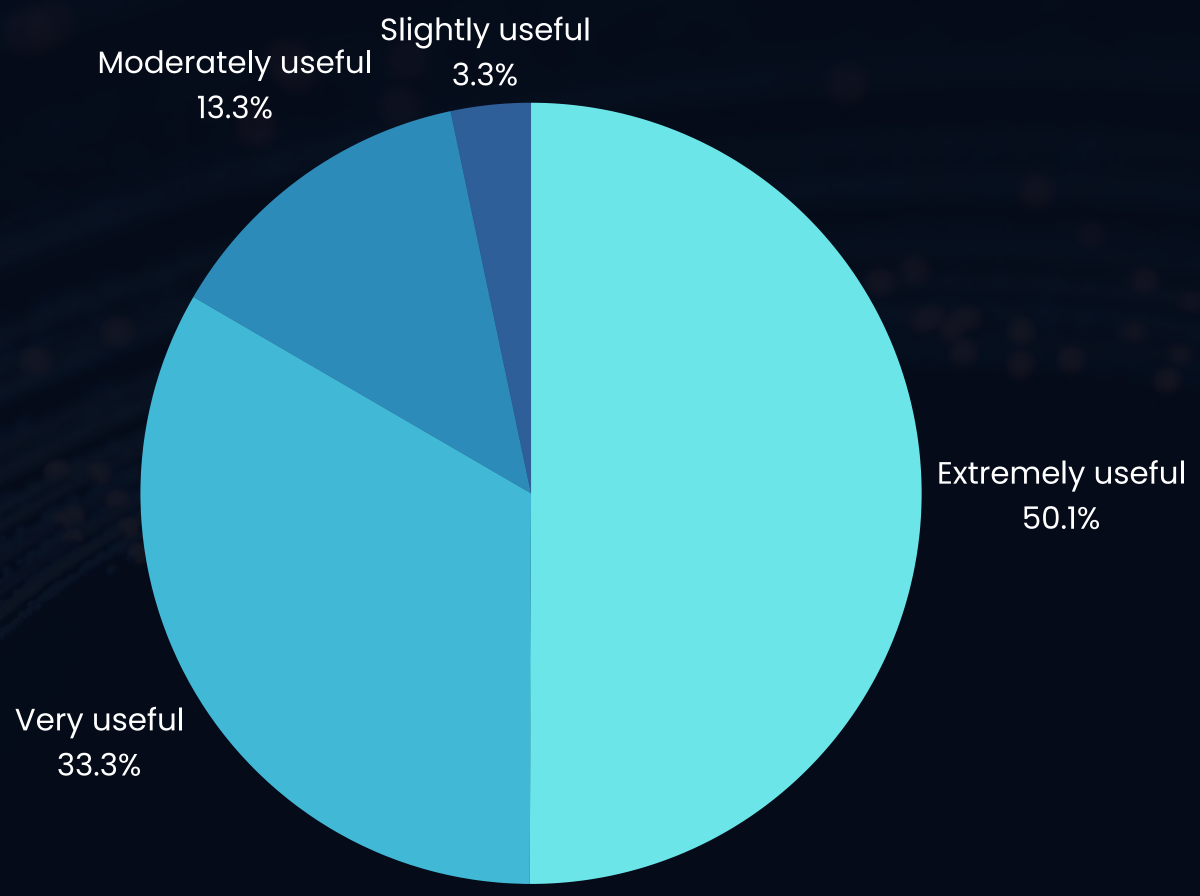
Strong trust in multi-level threat analysis

Users particularly value the integration of external threat sources such as VirusTotal and Google Safe Browsing.

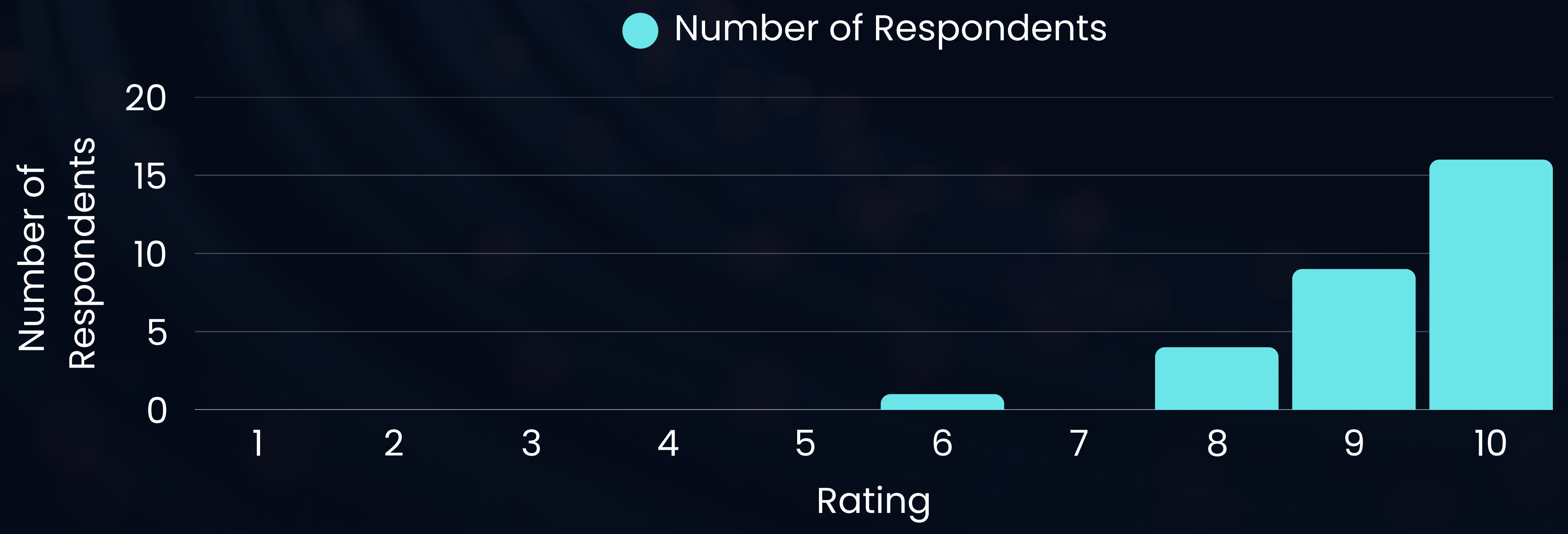
Safe Click is seen as useful and unique

The average overall user rating was 9.3 out of 10.
Over 93% of respondents indicated that they would be willing to install the extension.

How useful do you find the multiple security checks (Our Model Analysis, Google Safe Browsing, VirusTotal, DOM Analysis)?



Overall, how would you rate Safe Click extension on a scale of 1-10 (1 being poor, 10 being excellent)?



CONCLUSION

As part of this diploma project, an end-to-end smart system called Safe Click was developed and deployed to detect phishing URLs in real time. The solution integrates a Google Chrome browser extension with a Flask backend powered by a Random Forest machine learning algorithm.

ASTANA IT UNIVERSITY

DEMONSTRATION OF THE PRODUCT

ASTANA IT UNIVERSITY

DOWNLOAD PRODUCT

ASTANA IT UNIVERSITY

THANK YOU FOR ATTENTION!